Tele2 Content Billing Client Protocol
Content Billing Gateway

Version 4.45

History

Issue	Date	Reason for Changes
4.20	2008-09-01	Additional information about IP Number translation
4.21	2008-09-16	Minor corrections
4.22	2008-09-22	Incorporated previous document about ProviderTransactionID
4.23	2008-10-09	Add status codes for Limits, for future use
4.24	2008-10-15	Merge information from separate document concerning ContactInformation for Swedish CAs
4.25	2008-10-22	Add error code information for SOAP request
4.26	2008-11-12	Add additional status code information
4.27	2008-11-21	Minor change in 5.1.1.2
4.28	2009-01-15	Resending of same ProviderTransactionID after timeout expires (10 sec), modified text on TRANSLATEIP and buying process
4.29	2009-02-12	Correct small mistakes; define new Content Types and Status Codes
4.30	2009-02-25	Improved explanation of amount + VAT
4.31	2009-03-10	New Content Types & new Status Codes
4.32	2009-04-03	Details of TranslateIP XML; clarification of XtraData length
4.33	2009-04-06	Reinsert info about ErrorMessageFileds
4.34	2009-07-02	More detailed info about ProviderTransactionID and resending
4.35	2009-08-27	New content type requiring correct customer type to charge
4.36	2009-09-14	New content type
4.37	2009-09-29	Updated info about content description field
4.38	2009-11-27	New methods (Phone Model and IMEI)
4.39	2010-01-15	Clarification of Soap API
4.40	2010-01-18	Updated info about billing status 1004, new content_type 45
4.41	2010-01-26	New content types 46, 47, 48
4.42	2010-02-23	Updated content_type 26,30,32 and billing status 43, 997X, 998X
4.43	2010-04-14	Credit functionality (ReferenceID populated with a ProviderTransactionID of a previously charged transaction to credit)
4.44	2010-05-27	More detailed info about resending and ProviderTransactionID . New content_type 52 , positioning
4.45	2010-10-05	New content types 53, 81 (checking billing status without charging), New billing status codes 79 - 87

1. INTRODUCTION & SCOPE

1.1. The CBG Process

- End user contacts ContentProvider for purchase of an item or service.
- ContentProvider contacts Tele2 for charging using SOAP.
- Tele2 checks that the subscriber has sufficient funds for the purchase, makes a charge, and returns the result to the ContentProvider. See also “The Internal CBG Process” below.
- If the ContentProvider receives information that the subscriber has been charged, approves the purchase and deliver the item or service.
- Enduser receives item or service, and has been charged.

This document deals with the interaction between the ContentProvider and Tele2.

1.2. The Internal CBG Process

This receives data as defined in the request fields and returns data as defined in the response fields.

The response fields consist of a SOAP return code (RC) [typically 200], and a CBG status code in (CBGRESPONSE).

CBGRESPONSE consists of the CBG Internal status code (billing status) [often value 0 for ‘purchase OK’ – all values documented later herein] and a TransactionId (a unique value ALWAYS assigned to a transactions by the internal CBG process).

1.3. Summary of Tasks

This document describes the protocol to be used to communicate with the Tele2 Content Billing Gateway (CBG). The document is aimed for the content providers (CP) or content aggregators (CA) who shall initiate communication with the CBG.

1.4. Prerequisites and General Conditions

To be able to communicate with the CBG, the CA/CP has to sign a contract with Tele2. Tele2 will provide user-ID and password to be used for logging on to the CBG.

1.5. Versions, Versions, Versions

As from now, the *‘Version’* field of the Request Fields must be set to 208.
Where the version is set to 208, tags for ProviderTransactionID, ReferenceId and XtraData shall be populated.

Within the xml tagged data for XtraData, there is a subfield called *‘Version’* of 2 characters. Each version has its own format for the 98 characters of data which constitute the remainder of XtraData.

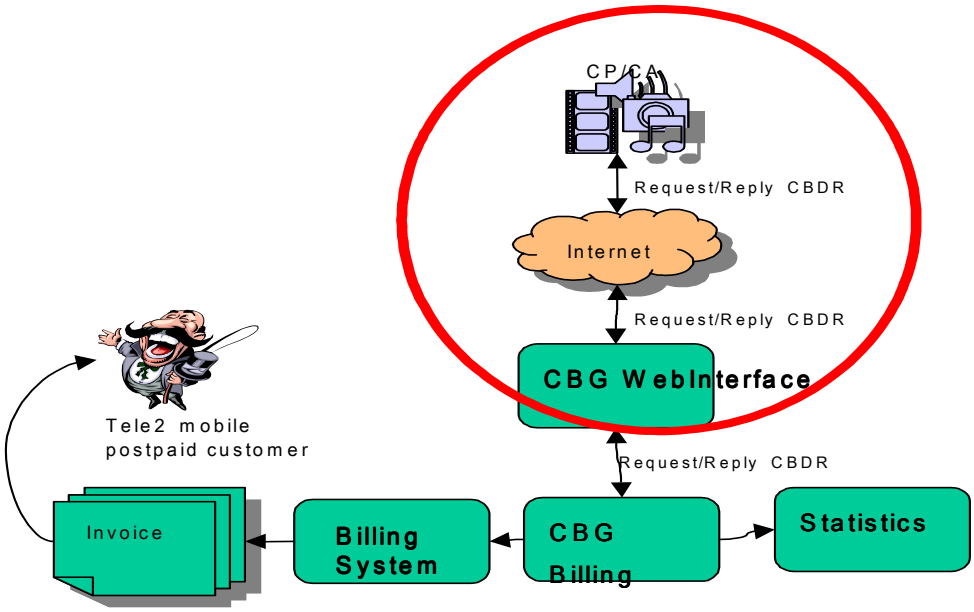
The previous value of the *‘Version’* field of the Request Fields was 203. This will continue to function for a ***while*** longer, but will eventually be discontinued – probably during 2009.

1.6. WSDL

A wsdl file for the development of the SOAP commands referred to in this document is available, and should be delivered to Content Providers together with this document.

2. INTERFACES

2.1. Architecture



2.2. Interface Description

2.2.1. Encoding Method

N/A

2.2.2. Transport Method

The transport method for CBG is TCP/IP.

Protocols
The data is carried using subsets of SOAP 1.1, over HTTPS.

More information regarding these protocol standards can be found at:

SOAP http://soapware.org
SOAP http://www.w3.org/TR/SOAP/
XML http://www.w3.org/TR/2000/REC-xml-20001006
HTTP http://www.w3.org/Protocols/rfc2616/rfc2616.html
SSL http://www.openssl.org

2.2.3. Description

The CA/CP client sends a Request message using this protocol and receives a message in the form of either a Response message or an Error message.

A Response message is sent back for all messages passing basic field control.
A Response message contains a transaction-ID which identifies the transaction in the Content Billing application and it should always be saved by the client along other transaction data for problem resolving etc.
A Response message also contains a status indicator indicating 3 possible outcomes:

- The transaction was committed
- The transaction was rejected. Something in the transaction data did not pass a control, e.g. authentication failed. A request receiving this kind of response should not be resent.
- The transaction failed. The transaction passed controls, but a dynamic fault occurred. A request receiving this kind of response may be resent.

An Error message is sent back when there are problems in the protocol format or whenever there is a communication problem in the backend network.

If you maintain a cookie named “Apache” you may only use that cookie for one specific user (see below). In this case the login.user and login.password fields may be left out for subsequent requests.

2.2.4. To keep in mind

Status 3 or 8 indicates that the Anumber is no longer a Tele2 subscriber, and shall be removed from any existing database immediately. Failure to do so will result in charging the wrong subscriber. Non-compliance can in extreme cases be grounds for termination of the CBG Agreement.

Check with the local Tele2 content manager what info should be put in the ContentDescription field. Additional information could be information about the purchased content.

For subscription services the billing requests shall be portioned in order to avoid congestion on the CBG.

ContentType in the billing request shall be set according to access used for delivering the Content.

2.2.5. Data Fields

2.2.5.1. Request Fields

Tag	Value	Mandatory	Description
Username	<string(6,64)>	yes	Content Providers login user
Password	<string(5,64)>	yes	Content Providers login password
Version	<Unsigned Int>	yes	203 or 208 are suported protocol versions
ContentType	<Unsigned Int>	yes	See. CBG protocol documentation.
Currency	<Unsigned Int>	yes	See. CBG protocol documentation.
Amount	<Unsigned Int>	yes	See. CBG protocol documentation.

Tag	Value	Mandatory	Description
VAT	<Unsigned Int>	yes	See. CBG protocol documentation.
OriginatingCustomerID	<string(5,20)>	yes	Phone Number in International format
ContentDescription	<string(0,41)>	yes	Content description to be displayed on the customer invoice. Check with local Tele2 content manager how to populate this field. Note, different length for different countries.
ProviderTransactionID	<Unsigned Int>	yes in 208	Values 1-2147483647, then back to 1.
ReferenceID	<Unsigned Int>	yes in 208	0 , if normal charging transaction. If a credit transaction, put ProviderTransactionID of a previously charged transaction to credit.
XtraData	<string(0,100)>	yes in 208	See CBG protocol XTRADATA documentation.

Instructions for populating these fields:

Tag	Type	Description
Username	String	Set to the username as given by Tele2
Password	String	Set to the password as given out by Tele2
Version	Integer	Should be set to 208. The previous value of 203 will function for a while longer
ContentType	Integer	A type indicator for statistic reports. The only allowed ContentTypes without special approval from Product Manager at Tele2 are: 0 – MMS Content 1 – WAP Content 2 – SMS Content/WAP-push 3 – MMS Traffic 4 – WEB Content 5 – Lottery 6 – WiFi 7 – MO Content 8 – Personal Ring Back Tone 9 – Royalty related content 10 – MMS within the Tele2 network 11 – MMS to other operators 12 – MMS to e-mail addresses 14 – Tickets 15 – Business 16 – Age check 16 yrs (Norway only) 17 – reserved for special use 18 – Age check 18 yrs (Norway only) 19 – Charity 20 – Tele2 internal, special billing (0% vat only) 21 – Parking 22 – Tele2 internal, mobile IM 25 – Tele2 internal, special billing (5% vat only) 26 – Tele2 internal, customer check, no charging 30 – Tele2 internal, customer info PRE/POST, no charging 31 – Mobile Portal Content 32 – Tele2 internal, limit info, no charging 34 – Tele2 internal, Mobile (Handset) Backup

		35 – Tele2 internal, Limits related Voice 36 – Tele2 internal, Refill functions related 38 – Ring Back Tone, charging for prepaid customer type 39 – Ring Back Tone, charging for postpaid customer type 40 – Charging for prepaid customer type only 41 – Charging for postpaid customer type only 42 – Auction 43 – Vending machine 45 – Tele2 internal, non-content product 46 – Tele2 internal, Limits related Data 47 – Tele2 internal, Limits related MBB 48 – Tele2 internal, Limits related SMS 50 – Adult content (Switzerland only) 51 – Directory enquiries 52 – Positioning 53 – Security product 80 – Test (for CBG SIT environment only) 81 – Check billing status, no charging 100 series: not in use 200 series: not in use 300 series: not in use 400 series: not in use 500 series: reserved for internal use. 600 series: not in use
Currency	Integer	A currency indicator. Allowed numeric values are 1 – SEK (Sweden) 2 – NOK (Norway) 3 – DKK (Denmark) 4 – EEK (Estonia) (valid up to and including 2010-12-31) 5 – EUF (Finland) 6 – EUH (Holland) 7 – EUL (Luxembourg) 8 – LVL (Latvia) 9 – LTL (Lithuania) 10 – EUA (Austria) 11 – RUB (Russia) 12 – USD (Russia) 13 – HRK (Croatia) 14 – CHF (Switzerland) 15 – EUE (Estonia) (valid from 2011-01-01)
Amount	Integer	Price including VAT expressed in local currency subunits, e.g.Swedish öre, Eurocents etc. Ask your local Tele2 product manager. EG. To charge 5 €, set tag “amount” to 500
VAT	Integer	A value in hundreds of percent indicating the VAT. Defaults to 2500 if not set. Ask your local Tele2 Product Manager for correct VAT. EG for 22% VAT, set tag “VAT” to 2200
OriginatingCustomerId	String	This shall be in MSISDN format (which is 15 digit maximum). MSISDN = CC + NDC + SN (Country Code + National Destination Code + Subscriber Number) Add an “00” International prefix. For example 0046708991199.The field is defined for a maximum of 20 characters, although only 17 will be used at present.
ContentDescription	String	Content description to be displayed on the customer

		invoice. Check with local Tele2 content manager how to populate this field. For Sweden: Max 41 chars. For Estonia, Latvia, Lithuania, Croatia this is restricted to 18 characters – a limitation imposed by the billing system. For Sweden the last 6 characters of the content_description field (static pos 36 (included) – pos 41 (included)) can be used by content_aggregators to populate a code (“contact_id”) referring to a contact_info displayed on the invoice.
ProviderTransactionID	Integer	A unique transaction ID from the CP in question. Although defined as unsigned integer, use positive range of signed int 1-2147483647.After 2147483647, restart at 1
ReferenceID	Integer	0 =>”Normal purchase” or a value referring to a previously sent ProviderTransactionId to be credited.
XtraData	String	Data from the CP. Exactly 100 characters. See definition below.

2.2.5.2. Response Fields

Tag	Value	Description
RC	Signed	Return code 200-600
CBGRESPONSE	Dict	Response from CBG with TransactionID and Status

Note 1:
The ordering of the key-value pairs in the response message is not pre-defined. A Content Provider **can not** assume that the XML tag with key = “RC” will be printed before the tag with key = “CBGRESPONSE”. This means that the Content Provider **must** parse the response message **both** with respect to the **tag/key** and its **value**.

Note 2:
The **only** key-value pairs allowed to implement **automated functionality** on are the ones **documented here** in this paragraph. **Any other** possible key-value pair that is sent inside the resonse message may be used for logging purposes, but must be treated as **optional and informational only**.

Instructions for interpreting the CBGRESPONSE fields:

Tag	Type	Description
TransactionId	String	Identification of the transaction. Max 30 chars.
Status	Integer	Indication of transaction status. See section 5 ”Transaction Status”.

Note:
The key-value pairs inside the CBGRESPONSE dict has the same generic requirements as the key-valuepairs in the overall response message. There is no guarantee that the transactionid tag is printed

before the status tag. Also, any other key/tag in this dict response and not defined here in this paragraph must be parsed and treated as optional and informational only (i.e. not used in automated processing but possibly for debugging and logging purposes only).

2.2.5.3. Translate IP Request Parameters

Tag	Value	Mandatory	Description
Username	<string(6,64)>	yes	Content Providers login user
Password	<string(5,64)>	yes	Content Providers login password
Version	<Unsigned Int>	yes	203 or 208 are 9upported protocol versions
OriginatingCustomerIP	<string(7,15)>	yes	IP address on dotted format

2.2.5.4. Get Phone Model Request Parameters

Tag	Value	Mandatory	Description
Username	<string(6,64)>	yes	Content Providers login user
Password	<string(5,64)>	yes	Content Providers login password
Version	<Unsigned Int>	yes	208 is 9upported protocol version
MSISDN	String	yes	Same as “OriginatingCustomerId” in Purchase request specification

2.2.5.5. Get IMEI Request Parameters

Tag	Value	Mandatory	Description
Username	<string(6,64)>	yes	Content Providers login user
Password	<string(5,64)>	yes	Content Providers login password
Version	<Unsigned Int>	yes	208 is 9upported protocol version
MSISDN	String	yes	Same as “OriginatingCustomerId” in Purchase request specification

2.2.5.6. Error Message Fields

Tag	Type	Description
rc	nonNegativeInteger	Return Code
error_message	string	Error Message
error_code	String/long/integer	Error Code

3. XtraData Field

3.1. Instructions for coding

Users of CBG shall deliver 100 characters of data in this tag.

Whilst we realise the difficulty of the task, we would nonetheless appreciate it if these 100 characters were populated using the documentation provided.

3.2. XtraData Field Version 01

SubField Name	Type	Description	
Version	String	Length 2	Value '01'.
Delimiter1	String	Length 1	Value ';'.
ContactID	String	Length 6.	
Delimiter2	String	Length 1	Value ';'.
ShortCode	String	Length 20.	
Delimiter3	String	Length 1	Value ';'.
Volume	String	Length 6.	
Delimiter4	String	Length 1	Value ';'.
DistributionType	String	Length 3	
Delimiter5	String	Length 1	Value ';'.
PurchaseType	String	Length 3.	
Delimiter6	String	Length 1	Value ';'.
DeliveryType	String	Length 3.	
Delimiter7	String	Length 1	Value ';'.
CategoryType	String	Length 3.	
Delimiter8	String	Length 1	Value ';'.
CategorySubType	String	Length 26.	
Delimiter9	String	Length 1	Value ';'.
ReasonCode	String	Length 2	
Delimiter9	String	Length 1	Value ';'.
PINCode	String	Length 8	
Delimiter10	String	Length 1	Value ';'.
Unused	String	Length 7.	

3.3. XtraData Field Version 02

For Tele2 internal use only (specifically MMS).

SubField Name	Type	Description	
Version	String	Length 2	Value '02'.
Filler	String	Length 98	Value undefined Recommend that this is populated with spaces or zeros.

4. New parameters in protocol 208

4.1 ProviderTransactionID

To increase the transaction security in the purchasing process, a new field ProviderTransactionId has been added in the protocol.

The Content Provider must set a unique ProviderTransactionId in every billing request. Start at value 1 and increase this incrementally by 1 up to 2147483647, after 2147483647, restart at 1.

If resending a transaction, always use the SAME ProviderTransactionID as the original, unless the status code (billing status) in the CBGRESPONSE part states that it is ok to increase.

Ie, in case the content provider does not receive a reply from the CBG, the transaction billing status is not known to the content provider, but the transaction can have been billed in the CBG. A resend (after transaction timeout period has expired (10 sec)) with the same ProviderTransactionID (duplicates are saved for 24 hours) can be done. If transaction was already processed, the resend will be rejected as a duplicate. Duplicates are checked per account (ContentProviderID).

Purpose:
The purpose of the ProviderTransactionID is to avoid double charging when the initial transaction failed due to technical problems or reply is lost and the content provider gets no reply on a request (ie status -32400 or it is timeouted at the content provider side). If billing status is unknown the transaction can be re-sent with the same ProviderTransactionId as the original request. If a duplicate ProviderTransactionID is received, CBG will reply status 44, 46 or possibly 999X (depending on the functionality in use), the X will represent the status of the original request with the same ProviderTransactionId. Take action according to the response of the original request.
OBS, if CBG status is received and status advices to increase, a new ProviderTransactionID must be used, otherwise the resending will be regarded as a duplicate.

Example:
1. Original request: ProviderTransactionId = 15 is sent and the content provider gets no reply from CBG (transaction is timeouted at content provider side ie http error -32400).
2. The Content_provider resends the request after timeout period expired (10 sec) with the SAME ProviderTransactionID = 15.
3. Possible answers:
    a. Reply is status = 9990, == > original request with ProviderTransactionId = 15 was status 0. Purchase OK, customer was charged successfully and content should be delivered.
    b. Reply is status = 9999, == > original request with ProviderTransactionId = 15 was status 9. Customer account balance is to low, reject.
    c. Reply is status = 44 , == > original request with ProviderTransactionId = 15 was already processed, but status functionality is not active and original status cannot be shown. Consider as Failed, reject.
    d. Reply is status = 46 , == > original request with ProviderTransactionId = 15 was already processed, status functionality is active but could not find original status. Consider as Failed, reject.
    e. No reply, transaction is timeouted again at content provider (ie http error -32400), == > resend again after timeout period expired (10 sec) with the SAME ProviderTransactionId = 15. Consider as Failed, resend max 3 times. Wait 10 sec between.

Recommended testing:
1. Send a request with a ProviderTransactionId, wait for the reply.
2. Resend request with the same ProviderTransactionId and check that the reply is 44, 46 or 999X.

If you receive 999X, then check that X matches the reply status of the original request.

4.1.1 Billing status check with ProviderTransactionID

For cases when billing status needs to be checked of a previously sent transaction without re-trying charging, send content_type = 81 and the SAME ProviderTransactionID as previously sent transaction to be checked (for security reasons set amount = 0). CBG will reply status 44, 46 , 87 or 999X (depending on the functionality in use), the X will represent the status of the original request with the same ProviderTransactionId.

This can be used for cases when no reply is received from CBG and the billing status is unknown or when customer has required charging approval and billing status is pending customer response.

4.2 ReferenceID

For normal charging, set ReferenceId = 0.
For crediting a previously charged transaction, set ReferenceID to the value of a previously sent ProviderTransactionID to credit (same A-number, content_type etc should be used as in the previously charged transaction).
For a credited transaction a negative revenue share (based on the credited amount) is added to the content provider revenue share account.

Example:

1. Original charge request: Amount = 100, ProviderTransactionId = 15 AND ReferenceID = 0 is sent, the content provider gets reply Status = 0 == > customer is charged.
2. To credit the charge above, Content_provider sends a request with a new ProviderTransactionID = 16 (increased to a not previously used value) AND ReferenceID = 15 (referring to the charge transaction to credit) and Amount <= 100 (full or partial amount of original charge can be credited, only one credit transaction per original charge transaction is allowed), if content provider gets reply Status = 0 == > customer is credited.

Check with your content manager in what countries the credit function is active. Currently only available for Sweden.

5. Examples using SOAP 1.1 (including ProviderTransactionID and XtraData)

5.1.1. Request


```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/1999/XMLSchema"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<T2api:Call xmlns:T2api="urn:/T2api/Proto/Soap">
 <T2api:request>
   <T2api:url>CBG</T2api:url>
   <T2api:method>Purchase</T2api:method>
   <T2api:kwargs>

<T2api:item><T2api:key>Version</T2api:key><T2api:valueUnsigned>208</T2api:valueUnsigned></T2api:it
em>

<T2api:item><T2api:key>ContentType</T2api:key><T2api:valueUnsigned>1</T2api:valueUnsigned></T2ap
i:item>

<T2api:item><T2api:key>Currency</T2api:key><T2api:valueUnsigned>1</T2api:valueUnsigned></T2api:ite
m>

<T2api:item><T2api:key>Amount</T2api:key><T2api:valueUnsigned>100</T2api:valueUnsigned></T2api:it
em>

<T2api:item><T2api:key>VAT</T2api:key><T2api:valueUnsigned>2500</T2api:valueUnsigned></T2api:ite
m>

<T2api:item><T2api:key>OriginatingCustomerId</T2api:key><T2api:valueString>0046704123456</T2api:va
lueString></T2api:item>

<T2api:item><T2api:key>username</T2api:key><T2api:valueString>K010101</T2api:valueString></T2api:it
em>

<T2api:item><T2api:key>password</T2api:key><T2api:valueString>SecretPassword</T2api:valueString></
T2api:item>

<T2api:item><T2api:key>ContentDescription</T2api:key><T2api:valueString>ProviderDefinedText</T2api:v
alueString></T2api:item>

<T2api:item><T2api:key>ProviderTransactionId</T2api:key><T2api:valueUnsigned>1234</T2api:valueUnsi
gned></T2api:item>

<T2api:item><T2api:key>ReferenceID</T2api:key><T2api:valueUnsigned>0</T2api:valueUnsigned></T2ap
i:item>
    <T2api:item><T2api:key>XtraData</T2api:key><T2api:valueString>Provider defined
text</T2api:valueString></T2api:item>
   </T2api:kwargs>
 </T2api:request>
</T2api:Call></SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

5.1.2. Response

Example 1)


```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/1999/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/1999/XMLSchema" xmlns:T2api="urn:/T2api/Proto/Soap">
<SOAP-ENV:Body>
<T2api:Response>
<T2api:rc>200</T2api:rc>
<T2api:data>
<T2api:item><T2api:key>rc_message</T2api:key><T2api:valueString></T2api:valueString></T2api:item>
<T2api:item><T2api:key>CBGRESPONSE</T2api:key>
<T2api:valueDict>
<T2api:item><T2api:key>TransactionId</T2api:key><T2api:valueString>123456789</T2api:valueString></T
2api:item>
<T2api:item><T2api:key>Status</T2api:key><T2api:valueUnsigned>0</T2api:valueUnsigned></T2api:item
>
</T2api:valueDict>
</T2api:item>
<T2api:item><T2api:key>rc_string</T2api:key><T2api:valueString>Success</T2api:valueString></T2api:ite
m>
</T2api:data></T2api:Response></SOAP-ENV:Body></SOAP-ENV:Envelope>
```


Example 2)


```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/1999/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/1999/XMLSchema" xmlns:T2api="urn:/T2api/Proto/Soap">
<SOAP-ENV:Body>
<T2api:Response>
<T2api:rc>200</T2api:rc>
<T2api:data>
<T2api:item><T2api:key>CBGRESPONSE</T2api:key>
<T2api:valueDict>
<T2api:item><T2api:key>TransactionId</T2api:key><T2api:valueString>123456789</T2api:valueString></T
2api:item>
<T2api:item><T2api:key>Status</T2api:key><T2api:valueUnsigned>0</T2api:valueUnsigned></T2api:item
>
</T2api:valueDict>
</T2api:item>
</T2api:data></T2api:Response></SOAP-ENV:Body></SOAP-ENV:Envelope>
```


Example 3)


```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/1999/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/1999/XMLSchema" xmlns:T2api="urn:/T2api/Proto/Soap">
<SOAP-ENV:Body>
<T2api:Response>
<T2api:rc>200</T2api:rc>
<T2api:data>
<T2api:item><T2api:key>CBGRESPONSE</T2api:key>
<T2api:valueDict>
<T2api:item><T2api:key>Status</T2api:key><T2api:valueUnsigned>0</T2api:valueUnsigned></T2api:item
>
```

<T2api:item><T2api:key>TransactionId</T2api:key><T2api:valueString>123456789</T2api:valueString></T2api:item>
</T2api:valueDict>
</T2api:item>
<T2api:item><T2api:key>rc_string</T2api:key><T2api:valueString>Success</T2api:valueString></T2api:item>
<T2api:item><T2api:key>rc_message</T2api:key><T2api:valueString></T2api:valueString></T2api:item>
</T2api:data></T2api:Response></SOAP-ENV:Body></SOAP-ENV:Envelope>

Note: These are examples. There is no guarantee that the items will be delivered in this order; all key/value pairs must be parsed properly. Any non-documented key/value pair must be treated as informational/optional data only (e.g. used for troubleshooting/logging purposes).

5.1.3. Translate IP XML : Request

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/1999/XMLSchema" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<T2api:Call xmlns:T2api="urn:/T2api/Proto/Soap">
 <T2api:request>
   <T2api:url>CBG</T2api:url>
   <T2api:method>TranslateIP</T2api:method>
   <T2api:kwargs>

<T2api:item><T2api:key>Version</T2api:key><T2api:valueUnsigned>208</T2api:valueUnsigned></T2api:item>

<T2api:item><T2api:key>OriginatingCustomerIP</T2api:key><T2api:valueString>123.45.67.87</T2api:valueString></T2api:item>

<T2api:item><T2api:key>username</T2api:key><T2api:valueString>K010101</T2api:valueString></T2api:item>

<T2api:item><T2api:key>password</T2api:key><T2api:valueString>SecretPassword</T2api:valueString></T2api:item>
   </T2api:kwargs>
 </T2api:request>
</T2api:Call></SOAP-ENV:Body>
</SOAP-ENV:Envelope>

5.1.4. Translate IP XML : (Normal, rc 200) Response


```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/1999/XMLSchema-
instance" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<T2api:Response>
   <T2api:rc>200</T2api:rc>
   <T2api:data>

<T2api:item><T2api:key>OriginatingCustomerId</T2api:key><T2api:valueString>00460704123456</T2api:v
alueString></T2api:item>
   </T2api:data>
</T2api:Response>
</SOAP-ENV:Body></SOAP-ENV:Envelope>
```


Note: This is an example. There is no guarantee that the items will be delivered in this order; all key/value pairs must be parsed properly. Any non-documented key/value pair must be treated as informational/optional data only (e.g. used for troubleshooting/logging purposes).

5.1.5. Translate IP XML : (rc 452) Response


```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema" xmlns:T2api="urn:/T2api/Proto/Soap">
<SOAP-ENV:Body>
<T2api:Response>
<T2api:rc>452</T2api:rc>
<T2api:data>
<T2api:item><T2api:key>error_message</T2api:key><T2api:valueString>ObjectNotFound</T2api:valueStri
ng></T2api:item>
<T2api:item><T2api:key>error_code</T2api:key><T2api:valueString>ObjectNotFound</T2api:valueString><
/T2api:item>
</T2api:data>
</T2api:Response>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


5.1.6. Get Phone Model : Request


```
<?xml version=" 1.0"  encoding=" UTF-8"  ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=" http://schemas.xmlsoap.org/soap/envelope/"  xmlns:SOAP-
ENC=" http://schemas.xmlsoap.org/soap/encoding/"  xmlns:xsd=" http://www.w3.org/1999/XMLSchema"
xmlns:xsi=" http://www.w3.org/1999/XMLSchema-instance"  SOAP-
ENV:encodingStyle=" http://schemas.xmlsoap.org/soap/encoding/"  >
<SOAP-ENV:Body>
```

```
<T2api:Call xmlns:T2api=" urn:/T2api/Proto/Soap" >
 <T2api:request>
  <T2api:url>CBG</T2api:url>
  <T2api:method>GetPhoneModel</T2api:method>
  <T2api:kwargs>
   <T2api:item><T2api:key>Version</T2api:key><T2api:valueUnsigned>208</T2api:valueUnsigned></T2api:item>
   <T2api:item><T2api:key>MSISDN</T2api:key><T2api:valueString>0046704123456</T2api:valueString></T2api:item>
   <T2api:item><T2api:key>username</T2api:key><T2api:valueString>K123456</T2api:valueString></T2api:item>
   <T2api:item><T2api:key>password</T2api:key><T2api:valueString>VerySecretPassword</T2api:valueString></T2api:item>
  </T2api:kwargs>
 </T2api:request>
</T2api:Call></SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


5.1.7. Get Phone Model : (Normal, rc = 200) Response


```
<?xml version=" 1.0" encoding=" UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=" http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC=" http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi=" http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd=" http://www.w3.org/1999/XMLSchema" xmlns:T2api=" urn:/T2api/Proto/Soap" >
<SOAP-ENV:Body>
<T2api:Response>
<T2api:rc>200</T2api:rc>
<T2api:data>
<T2api:item><T2api:key>PHONEMODEL</T2api:key><T2api:valueString>SonyEricsson S500i/SonyEricsson
S500c</T2api:valueString></T2api:item>
<T2api:item><T2api:key>rc_message</T2api:key><T2api:valueString></T2api:valueString></T2api:item>
<T2api:item><T2api:key>rc_string</T2api:key><T2api:valueString>Success</T2api:valueString></T2api:item>
</T2api:data></T2api:Response></SOAP-ENV:Body></SOAP-ENV:Envelope>
```


Note: This is an example. There is no guarantee that the items will be delivered in this order; all key/value pairs must be parsed properly. Any non-documented key/value pair must be treated as informational/optional data only (e.g. used for troubleshooting/logging purposes).

5.1.8. Get Phone Model : (rc = 452) : Response


```
<?xml version=" 1.0" encoding=" UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=" http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC=" http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi=" http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd=" http://www.w3.org/1999/XMLSchema" xmlns:T2api=" urn:/T2api/Proto/Soap" >
<SOAP-ENV:Body>
<T2api:Response>
<T2api:rc>452</T2api:rc><T2api:data>
<T2api:item><T2api:key>rc_string</T2api:key><T2api:valueString>ObjectNotFound</T2api:valueString></T2api:item>
<T2api:item><T2api:key>rc_message</T2api:key><T2api:valueString>Subscriber not found</T2api:valueString></T2api:item>
```

<T2api:item><T2api:key>error_message</T2api:key><T2api:valueString>Subscriber not found</T2api:valueString></T2api:item>
<T2api:item><T2api:key>error_code</T2api:key><T2api:valueString>ObjectNotFound</T2api:valueString></T2api:item>
</T2api:data></T2api:Response>
</SOAP-ENV:Body></SOAP-ENV:Envelope>

Note: This is an example. There is no guarantee that the items will be delivered in this order; all key/value pairs must be parsed properly. Any non-documented key/value pair must be treated as informational/optional data only (e.g. used for troubleshooting/logging purposes).

5.1.9. Get IMEI : Request

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/1999/XMLSchema" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<SOAP-ENV:Body>
<T2api:Call xmlns:T2api="urn:/T2api/Proto/Soap">
<T2api:request>
<T2api:url>CBG</T2api:url>
<T2api:method>GetIMEI</T2api:method>
<T2api:kwargs>
<T2api:item><T2api:key>Version</T2api:key><T2api:valueUnsigned>208</T2api:valueUnsigned></T2api:item>
<T2api:item><T2api:key>MSISDN</T2api:key><T2api:valueString>0046704123456</T2api:valueString></T2api:item>
<T2api:item><T2api:key>username</T2api:key><T2api:valueString>K123456</T2api:valueString></T2api:item>
<T2api:item><T2api:key>password</T2api:key><T2api:valueString>NEVERshowthispassEVER</T2api:valueString></T2api:item>
</T2api:kwargs>
</T2api:request>
</T2api:Call></SOAP-ENV:Body>
</SOAP-ENV:Envelope>

5.1.10. Get IMEI : (Normal, rc = 200) Response

<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns:xsd="http://www.w3.org/1999/XMLSchema" xmlns:T2api="urn:/T2api/Proto/Soap">
<SOAP-ENV:Body>
<T2api:Response><T2api:rc>200</T2api:rc>
<T2api:data>
<T2api:item><T2api:key>IMEI</T2api:key><T2api:valueString>354059020628620F</T2api:valueString></T2api:item>
<T2api:item><T2api:key>rc_message</T2api:key><T2api:valueString></T2api:valueString></T2api:item>
<T2api:item><T2api:key>rc_string</T2api:key><T2api:valueString>Success</T2api:valueString></T2api:item>

```
</T2api:data></T2api:Response></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Note: This is an example. There is no guarantee that the items will be delivered in this order; all key/value pairs must be parsed properly. Any non-documented key/value pair must be treated as informational/optional data only (e.g. used for troubleshooting/logging purposes).

5.1.11. Get IMEI : (rc = 452) Response

```
<?xml version=" 1.0" encoding=" UTF-8" ?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV=" http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC=" http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi=" http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd=" http://www.w3.org/1999/XMLSchema" xmlns:T2api=" urn:/T2api/Proto/Soap" >
<SOAP-ENV:Body>
<T2api:Response>
<T2api:rc>452</T2api:rc><T2api:data>
<T2api:item><T2api:key>rc_string</T2api:key><T2api:valueString>ObjectNotFound</T2api:valueString></T2api:item>
<T2api:item><T2api:key>rc_message</T2api:key><T2api:valueString>Subscriber not found</T2api:valueString></T2api:item>
<T2api:item><T2api:key>error_message</T2api:key><T2api:valueString>Subscriber not found</T2api:valueString></T2api:item>
<T2api:item><T2api:key>error_code</T2api:key><T2api:valueString>ObjectNotFound</T2api:valueString></T2api:item>
</T2api:data></T2api:Response>
</SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Note: This is an example. There is no guarantee that the items will be delivered in this order; all key/value pairs must be parsed properly. Any non-documented key/value pair must be treated as informational/optional data only (e.g. used for troubleshooting/logging purposes).

5.1.12. Return Codes

Return codes are set in the CBG webinterface and indicate if parsing of message was successful (200 = successful parsing), if so also the CBGRESPONSE part in the reply will be populated with CBG Status code (billing status) and CBG TransactionID (Tele2 billing TransactionID), see chapter 6.

If resending a transaction, always use the SAME ProviderTransactionID as the original, unless the status code (billing status) in the CBGRESPONSE part states that it is ok to increase.

HTTP Return Codes <T2api:rc>*</T2api:rc>	Web Standard Meaning	Content Provider Action
1XX	Informational Return Codes	
2XX	Successful Return Codes	
3XX	Redirection Return Code	
4XX	Client Error Return Codes	
5XX	Server Error Return Codes	

CBG Return Codes <T2api:rc>*</T2api:rc>	CBG Meaning	Content Provider Action
200	Success	CBGRESPONSE part in the reply will be populated with CBG Status code (billing status). Check billing status in chapter 6.
201	Partial	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
202	Accepted	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
203	Already Done	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
204	Created	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
400	UnknownURI	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
401	ReadOnlyURI	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
402	UnknownMethod	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
410	UnknownKey	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
411	ReadonlyKey	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
412	SubspaceNotAllowed	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
413	SubspaceNeeded	Resend max 3 times with the SAME ProviderTransactionID as the original.

		Investigate reason.
414	UnknownSubspace	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
415	DuplicateKey	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
416	ValueNotSet	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
420	ParameterUnknown	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
421	ParameterNeeded	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
422	ParameterSyntaxError	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
423	ParameterInvalid	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
424	ParameterLengthInvalid	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
425	ParameterIllegalCharacters	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
430	AuthenticationFailed	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
431	AuthorizationFailed	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
432	Suspended	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
433	Disabled	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
440	ClientNotAuthenticated	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
441	ClientNotAuthorized	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
442	UserNotAuthenticated	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
443	UserNotAuthorized	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
444	NotEncypted	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.

450	ValueUndefined	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
451	SearchCriteriaTooWide	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
452	ObjectNotFound	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
453	OperationNotAllowed	Resend max 3 times with the SAME ProviderTransactionID as the original. Investigate reason.
Server Errors		
500	Unavailable	Resend max 3 times with the SAME ProviderTransactionID as the original
501	Aborted	Resend max 3 times with the SAME ProviderTransactionID as the original
502	ProtocolError	Resend max 3 times with the SAME ProviderTransactionID as the original
503	Cluster	Resend max 3 times with the SAME ProviderTransactionID as the original
510	Declined	Resend max 3 times with the SAME ProviderTransactionID as the original
520	NotConfigured	Resend max 3 times with the SAME ProviderTransactionID as the original
521	NotImplemented	Resend max 3 times with the SAME ProviderTransactionID as the original
530	TransactionFailed: often indicates a syntax error in the XML or incompatibility between XML and wsdl (possible with respect to the defined value). Where this is difficult to resolve, test with the XML from this document, and change only that which needs to be changed.	Resend max 3 times with the SAME ProviderTransactionID as the original
531	OutOfResources	Resend max 3 times with the SAME ProviderTransactionID as the original
532	UnknownSession	Resend max 3 times with the SAME ProviderTransactionID as the original
533	Again	
534	Timeout	Resend max 3 times with the SAME ProviderTransactionID as the original
540	BackendUnavailablePermanently	Resend max 3 times with the SAME ProviderTransactionID as the original
541	BackendUnavailableTemporarily	Resend max 3 times with the SAME ProviderTransactionID as the original

5.2. Error Scenarios

5.2.1. FAQ

Question:
Do I need an application to connect to CBG?
Answer:
No! You cannot ”connect” to CBG. You send a XML-request over SSL, (HTTPS) to the web service and receive a reply in XML over SSL.

Question:
I cannot find the URL to send the billing requests to in this specification.
Answer:
You will receive it together with your Content Provider ID (CPID) and Password by ordinary mail.

Question:
I receive error code 3 ”Customer Does Not Exist”. I am sure the user is a Comviq/Tele2 customer.
Answer:
The correct format if the customer is ”00 Contry Code Mobile Phone number”.
E.g. if the user has mobile number 0704000000 the correct format in the request is 0046704000000

Question:
I receive error code 3 ”Customer Does Not Exist”. I am not sure the user is a Comviq/Tele2 customer.
Answer:
CBG is not intended to be used for identifying a customer’s operator. Any request to CBG receiving a “Customer does not exist”-reply should be followed by an immediate deletion of the A-number from any existing customer database. The reason for this is the possible risk of charging new Tele2 customers with A-numbers previously owned by users requesting the content. This risk is only likely to happen for subscription services.

Question:
I cannot access the CBG URL with my web browser. Why is that? **Answer:**
You have to use XML and not HMTL which means that a web browser will not suffice. Also remember to use POST instead of GET.

Question:
I cannot access the web service and I’m sure that I’m using XML over SSL. I do use POST and not GET.
Answer:
Please check User-ID and password.

Question:
I still cannot communicate with the CBG.
Answer:
Due to limited resources Tele2 can unfortunately not supply implementation support. Please read the manual thoroughly or consider hiring a consultant for the implementation.

Question:
Why do I receive status code 4 “Purchase within time restriction” when I try to charge multiple transactions of the same customer for the same purchase.
Answer:
You are not allowed to do that, for two reasons:
1. The Maximum limit is intended for each purchase and not for each transaction.
2. Tele2 wants to prevent accidental multiple transactions of customer by mistake.

CBG is not intended for billing of services that are priced above the set maximum amount.

Question:
Can I set up Concurrent connections against the Tele2 content billing gateway? **Answer:**
The content provider may set up a maximum of 5 connections against the Tele2 content billing gateway. The main reason for this is that this is the only way to send more then a couple of transactions per second. The secondary reason is that some transactions might take up to several seconds to perform, and if the content provider only has one connection towards the gateway, all other transactions would stall during this period. How this technically is done is completely up to the content provider.

The same session (cookie), can be used for all connections, or they can use different sessions for different connections, or provide the username password in all requests.

6. Billing Status

If message parsing in the CBG webinterface was successful (ReturnCode = 200), the CBGRESPONSE part in the reply will be populated with CBG Status code (billing status) and CBG TransactionID (Tele2 billing TransactionID). For a billing to be considered successful (ReturnCode must be 200), CBG status must be 0 and there must be a numeric value of the CBG TransactionID .

Status to CA/CP	StatusDescription	Purchase / Action
0	Purchase OK	OK
1	ContentProvider Does not exist	Reject
2	Amount is out of range (lower or higher than allowed for the content provider account).	Reject
3	Customer does not exist	Reject Remove the A-number immediately from any existing customer database. The A-number should not be resent to CBG unless the user has done a new request.
4	Purchase within time restriction. Customer must not be charged with multiple transactions for the same service.	Reject
5	Invalid routing information	Reject
6	To many requests in timeframe	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
7	To many pending IN requests	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
8	Customer does not exist in IN	Reject Remove the A-number immediately from any existing customer database. The A-number should not be resent to CBG unless the user has done a new request.
9	Customer account balance to low	Reject
10	IN read timeout during balance check	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
11	IN read timeout during withdrawal	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
12	Communication error before withdrawal	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.

13	Communication error during withdrawal	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
14	Invalid CHARGE ACCOUNT	Reject
15	Invalid VAT_PERCENTAGE	Reject
16	Invalid CURRENCY	Reject
17	IN Node not connected	Reject
18	The content provider account and customer are of different nationalities	Reject
19	Request currency differs from provider account currency	Reject
20	Invalid provider currency. The currency is not registered correctly on the provider account	Reject
21	Max number of outstanding CGB requests exceeded	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
22	Customer has requested barring service for MMS or Content or both MMS and Content.	Reject
23	Internal error	Reject
24	Internal error	Reject
25	Internal error	Reject
26	Customer account has either expired, or never been activated for the content service (no first call). For the latter case, inform customer to make a voice call to activate service, then retry ordering content.	Reject Remove the A-number immediately from any existing customer database. The A-number should not be resent to CBG unless the user has done a new request.
27	Internal error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
28	Internal error – payment broker	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
29	Internal error/timeout	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
30	Internal error – transaction already open	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
31	Internal error – transaction busy	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
32	Internal error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.

33	Internal error – possible overload	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
35	Internal error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
36	Customer under 16 yrs	Reject
37	Age check failed (no age data on customer or multiple rows)	Reject
38	Customer under 18 yrs	Reject
39	Customer info prepaid	Reject
40	Customer info prepaid	Reject
41	Customer info error	Reject
42	Customer info postpaid	Reject
43	Customer info : customer active	Reject
44	Duplicate found (ProviderTransactionID already used)	Reject
45	Duplicate function error	Reject
46	Duplicate found (ProviderTransactionID already used) but previous transaction status not found	Reject
47	Incoming queue timeout1	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
48	Incoming queue timeout2	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
50	Customer has requested barring service for adult content	Reject
51	Customer limit for Content exceeded	Reject
52	Customer limit for MMS exceeded	Reject
53	Customer limit for Volume exceeded	Reject
54	Customer not Active	Reject
55	Customer info – No limit activated	Reject
56	Customer info – No valid limit found	Reject
57	Customer info prepaid	Reject
58	Customer content Limit = 0	Reject
59	Customer info postpaid	Reject
60	Customer info postpaid	Reject
61	Not correct customer type for charging	Failed, customer exists, resend with correct content_type for charging this customer type and new ProviderTransactionID
62	Credit: CREDIT_AMOUNT_LARGER_THAN_CHARGE	Reject Set correct amount
63	Credit: SELECT_CREDIT_STATUS_FAILED	Reject

64	Credit: CREDIT_CONTENTTYPE_DIFFERENT_THAN_CHARGE	Reject Set correct content_type
65	Credit: VAT_DIFFERENT_THAN_CHARGE	Reject Set correct VAT
66	Credit: CURRENCY_DIFFERENT_THAN_CHARGE	Reject Set correct Currency
67	Credit: ORIGINAL_CHARGE_TRANSACTION_NOT_SUCCESSFUL	Reject
68	Credit: CustomerType_DIFFERENT_THAN_CHARGE	Reject
69	Credit: CUSTOMER_DIFFERENT_THAN_CHARGE	Reject Set correct customer
70	Credit: TOO_OLD_TO_CREDIT	Reject
71	Credit: CONTENT_PROVIDER_NOT_ALLOWED_TO_CREDIT	Reject Contact content manager to get permission
72	Credit:	Reject
73	Credit: FAILED_NO_ORIGINAL_CHARGE_FOUND	Reject
74	Credit: CUSTOMER_TYPE_NOT_ALLOWED_TO_CREDIT	Reject
75	Credit: CREDIT_IS_DUPLICATE	Reject
76	Credit: CREDIT_DUPLICATE_CHECK_TABLE_ERROR	Reject
77	Credit: CREDIT_IS_DUPLICATE_SELECT_ORIGINAL_STATUS_FAILED	Reject

78	Credit: CREDIT_IS_DUPLICATE_SELECT_ORIGINAL_PROVIDERTRANSACTIONID_FAILED	Reject
79	Approval: pending customer approval	Pending customer approval, check billing status with new content_type 81 and SAME ProviderTransactionID. Wait 10 sec between resending check.
80	Approval: transaction rejected by customer	Reject
81	Approval: incorrect customer response	Reject
82	Approval: customer response time out	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
83	Approval: approval system error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
84	ProviderTransactionID not correctly populated	Reject
85	Internal error	Reject
86	Dup check: no duplicate found in duplicate transaction cache	Reject
87	Dup check: dup check functionality error	Reject
1001	Unknown MSISDN	Reject
1002	Internal error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
1003	Not Enough Credit	Reject
1004	Not Enough Credit	Reject
1005	Internal error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
1006	Internal error	Failed, resend max 3 times with new ProviderTransactionID. Wait 10 sec between resending.
995X	Credit Duplicate found (ReferenceID already used) This status code is used when duplicate_info function is active. Previous transaction status was X (part after the 995 prefix)	Do not resend. See X-part.
997X	Customer has MMS limit and current sum is X (MMS units)	Do not resend. See X-part.
998X	Customer has content limit and current sum is X (subunits of currency, ie Eurocents)	Do not resend. See X-part.

999X	Duplicate found (ProviderTransactionID already used). This status code is used when duplicate_info function is active. Previous transaction status was X (part after the 999 prefix)	Do not resend. See X-part.

Note 1. For cases when there are system or network problems, it can happen that a request or reply is lost and the content provider does not receive a reply from the CBG system. Then there might be error codes like:
-32400 or similar. This is an internal XML-RPC error set by the protocol itself at the content provider application side and not sent from the CBG. Action: Investigate your application why this is set. In case it is because of a timeout set on the content provider side, consider the transaction to be failed, resend max 3 times with the SAME ProviderTransactionID (since the previous transaction was probably lost) . Wait 10 sec between resending.
For other XML-RPC error codes like -32600, -32700 etc, see above, these codes are not sent from the CBG, please investigate the application at the content provider side.

Note 2. Errorcodes 1001 – 1006 are specific for Russian transactions.

6.1. Translation of an IP number to an A-number (eg WAP-traffic)

The customer internet access will determine what customer information is available for content billing.

If the customer connects via the WAP proxy and the content provider is trusted (has a WAP account at Tele2) the customers IP address in the cookie field “ip-address” can be used. If the content provider (CP) is not trusted (no WAP account at Tele2) no billing information is given, then the CP should contact Tele2 to sign up for a WAP account.

If the customer connects directly to the Internet, the CP has to use the source IP address.

The next step for the CP is to find out whether the customer is a Tele2/Comviq customer.
We exemplify two different ways to do this based on the IP address in examples below (see chapter 2.2.5).

The TRANSLATEIP function must be executed separately and before the normal CBG request.

Since, there is a small risc that the customer is disconnected during the TRANSLATEIP function call (before the A-number is established). To increase security in the purchasing process, CP must validate that the customer session is still the same after the TRANSLATEIP function has returned the A-number (eg display an “are you sure” for customer to confirm buying process). If the customer session has changed, the customer has been disconnected (the session was dropped and a new IP-address was distributed) and the A-number is not valid for this customer. If so, redirect the customer to the start of purchasing process and use the new IP address (and do a new TRANSLATEIP). If customer session is the same continue with CBG charging.

In order to translate an IP address to an A-number, the CP performs a similar request as the one to the CBG when asking to charge for content. They use the same login and password and the same URL. The only thing that differs is what arguments they send in the request and which request function they want to use, which is named TRANSLATEIP.

Request fields:
Field Name Type Description login.user String same user login as the normal CBG login.password String same user OriginatingCustomerIP String the IP address to be translated to an A-number

Output fields:

Field Name Type Description OriginatingCustomerId String the A-number to be used in the billing request

6.2. IP Ranges

The IP ranges that Tele2's customers use can change without notice. In order to know if a user is a Tele2 customer or if she/he is a customer to one of the other operators, the IP address has to be looked up in a database. There are two different kinds of databases that could be used for this. Choosing which of these to use is a decision left to the content provider.

The easiest way to perform the lookup is to make a reverse DNS query. If the reverse address contains tele2.xx or swipnet.se, the user is coming from the Tele2 network. For most content providers this would be the easiest way to implement the lookup. There are libraries that support this kind of lookup in virtually all programming environments.

The alternative way of doing the lookup is to make an inquiry to the RIPE database once a day and store all addresses belonging to Tele2 and swipnet in a local database. The RIPE database contains all IP addresses that have been assigned to Tele2. For more information visit http://www.ripe.net.

Currently not all Tele2 customers can be billed through the CBG. If the user is using a dial-up connection, the content provider will not be able to notice that it is not possible to bill the customer through the CBG until the request is sent to the CBG.

For details of XML see 4.1.1.3.

6.3. Capacity recommendation

Capacity recommendations for sending are:

Max 1 transaction/sec for subscription services and non capacity requiring services.
Max 1 transaction/sec for the age check functionality for Norway.
Max 5 transactions/sec for live voting services.